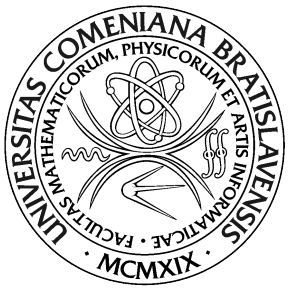
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



CONTEXTUALIZED LANGUAGE MODEL-BASED NAMED ENTITY RECOGNITION IN SLOVAK TEXTS

Diploma thesis

2021 Bc. Dávid Šuba

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

CONTEXTUALIZED LANGUAGE MODEL-BASED NAMED ENTITY RECOGNITION IN SLOVAK TEXTS

Diploma thesis

Study program:	Applied informatics
Branch of study:	2511 Applied informatics
Supervisor:	Mgr. Endre Hamerlik
Consultant:	Mgr. Marek Šuppa

Bratislava, 2021

Bc. Dávid Šuba

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Dávid Šuba
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Contextualized Language Model-based Named Entity Recognition in Slovak Texts
Rozpoznávanie pomenovaných entít kontextualizovanými jazykovými modelmi v slovenských textoch

Anotácia: Named Entity Recognition (NER) je jednou zo základných úloh spracovania prirodzeného jazyka (NLP), pričom najmodernejšie anglické prístupy všeobecne využívajú neurálne modely [1]. V súčasnosti dostupné klasifikátory NER pre slovenské texty sú buď systémy založené na pravidlách a slovnej zásobe, alebo využívajú viacjazyčné kontextuálne jazykové modely [2]. Oba vykazujú slabý výkon v porovnaní s jazykovo špecifickými hlboko kontextovými jazykovými modelmi, a to aj v jazykoch s nízkym počtom zdrojov, ako je napríklad slovenčina.

Cieľ:
1. Skontrolujte existujúce modely NER a súbory údajov, na ktoré sú trénovaní
2. Spresnite a rozšírte anotované (slovenské) súbory údajov pre NER
3. Využite výhody prenosového učenia medzi viacjazyčnými a slovenskými modelmi NER [3]

Literatúra:
[1] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.
[2] Kaššák, Ondrej - Kompan, Michal - Bieliková, Mária. Extrakcia pomenovaných entít pre slovenský jazyk. In ZNALOSTI 2012 : Sborník příspěvků 11 ročníku konference, Mikulov, hotel Eliška 14.-16. 10. 2012. Praha: Matfyzpress, 2012, pp. 52--61
[3] Rahimi, A., Li, Y., & Cohn, T. (2019). Massively multilingual transfer for NER. arXiv preprint arXiv:1902.00193

Vedúci: Mgr. Endre Hamerlik
Konzultant: Mgr. Marek Šuppa
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 19.03.2021

Dátum schválenia: 07.04.2021 prof. RNDr. Roman Ďurikovič, PhD.

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

garant študijného programu

.....

študent

.....

vedúci práce

I hereby declare that I have written this thesis by myself, only with help of referenced literature, under the careful supervision of my thesis advisor.

. . . . . .

Bratislava, 2021 Bc. Dávid Šuba

Acknowledgement

I want to thank....

Abstract

Named Entity Recognition (NER) is one of the fundamental tasks in Natural Language Processing (NLP), with English state-of-the-art approaches generally utilizing neural models. The currently available NER classifiers for Slovak texts are either rule- and vocabulary-based systems or employ multilingual Contextualized Language Models. Both of these show poor performance compared to the language-specific Deep Contextualized Language Models, even in low-resource languages, such as Slovak.

Keywords: named entity recognition, natural language processing, deep learning

Abstrakt

Rozpoznávanie pomenovaných entít (NER) je jedna zo základných úloh v spracovaní prirodzeného jazyka, kde najlepšie modely sú založené na neurónových sieťach. Aktuálne dostupné NER klasifikátory sú založené buď na pravidlách a slovnej zásobe alebo multi-jazykových kontextualizovaných jazykových modeloch. Obidva prístupy nedosahujú úroveň pre jazykovo špecifické hlboké kontextualizované jazykové modely dokonca ani v jazykoch s málo zdrojmi ako slovenčina.

Kľúčové slová: rozpoznávanie pomenovaných entít, spracovanie prirodzeného jazyka, hlboké učenie

Contents

Chapter 1

Introduction

1.1 Problem definition

The problem of Named Entity Recognition (NER) falls into a field called Natural Language Processing (NLP). NLP deals with natural texts - created by humans. In computer science we use programming languages with precisely defined syntax, markup languages, structured documents or databases where information is distinct and unambiguous. On the other hand we can say that information in natural texts is hidden and it requires a lot more knowledge and context to be able to retrieve it.

According to [1] NLP can be defined as follows: *"Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications"*. For better understanding there is a list of several most important NLP tasks in recent years:

- machine translation

- summarization
- natural language generation
- part of speech tagging
- named entity recognition
- sentiment analysis
- question answering
- search engines
- text predicting

1.1.1 Named Entity Recognition

NER is one of the most important NLP tasks. The task lies in recognizing named entities in text and classifying them into defined categories. The most common categories are Person (PER), Location (LOC), Organization (ORG) and Miscellaneous (MISC). However there can be many other categories depending on domain of texts. The figure 1.1 illustrates NER on simple sentence.

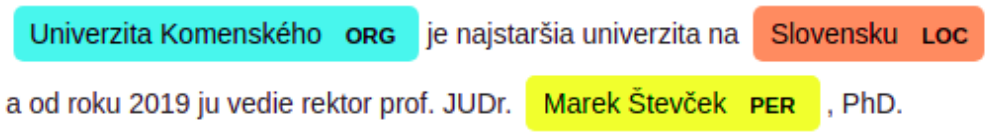


Figure 1.1: Illustration of Named Entity Recognition

The use case of NER lies mostly in the field of information extraction or so called data mining. An example usage can be automatic extraction

of data from unstructured CV. Identifying named entities would be the first and most important step in pipeline for retrieving important information from CVs. Another domains of NER usage can be recommendation systems, search engines or document classification.

1.2 Approaches to NER

This section is based on sources [2], [3], [4], [5] and [6].

Named entity recognition has been interesting for researchers for several decades.The task has been first defined at Sixth Message Understanding Conference (MUC-6) in 1996 but first publications aiming NER have been published around year 1991. During this period many different approaches for solving this task were introduced. We can divide most of them into several categories as shown in 1.2.

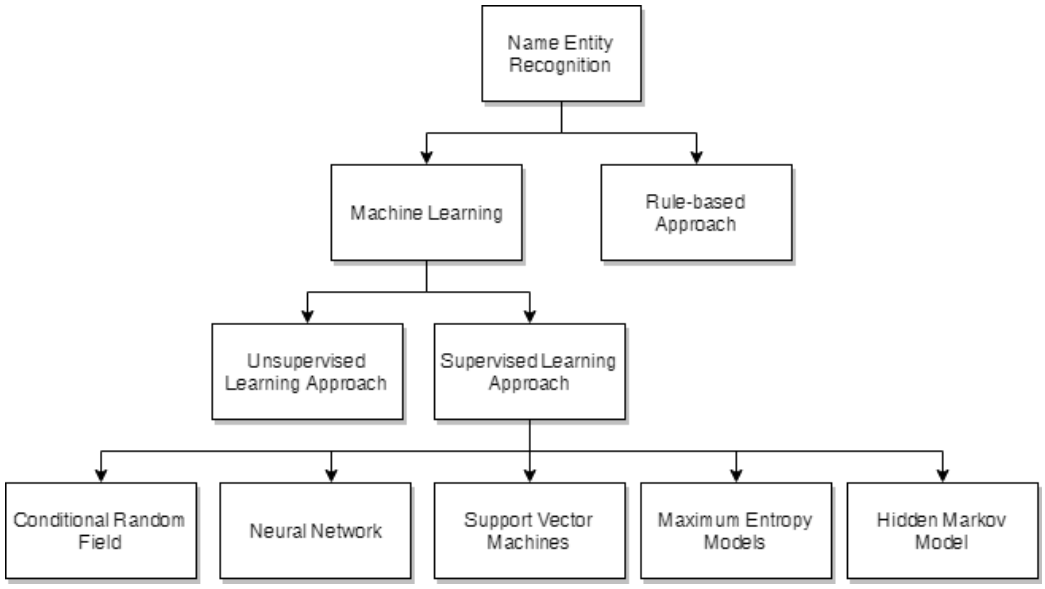


Figure 1.2: Hierarchical graph of main approaches to NER [7]

1.2.1 Rule bases approach

The most early and basic approach is based on hand-made rules. In early years it was dominating the field and nowadays it can still be used as a baseline for other methods to compare with or as an extension for more advanced ones.

It relies at many rules defined by human who is an expert in classifying named entities. For example the most basic and effective rule which almost anyone would think of when it comes to identifying named entities is capitalized first letter. When we think of Location, Person and Organization each of them has to start with capitalized letter and thus we have very effective and filtering rule for identification of possible named entities.
Other rules can exploit grammatical or lexical features such as part of speech. Many word patterns can be defined such as: there is always person entity after title (prof. Albert Einstein). Another method which is heavily used by rule-based systems are dictionaries and gazetteers of already known entities, e.g. we can use phone book to identify persons or scrape Wikipedia or some knowledge base which are structured and contain additional information and therefore it can be much easier to perform NER.

Although rule-based systems can be highly effective in some domains there are several major disadvantages. It requires a lot of manual work and people with non-trivial knowledge in the area. Also they are very domain specific and not easily transferable to other domains or even an language.

1.2.2 Machine learning approach

Because of the mentioned drawbacks of rule-based systems, researchers started leaning towards machine learning models. We can divide machine learning

into 2 main categories: Supervised learning and Unsupervised learning. Difference is that supervised learning algorithms learn from data that were annotated by humans. On the other hand idea behind unsupervised algorithms is to learn representations solely from data without any additional information. Despite some works targeting NER with unsupervised approach it is not very popular and supervised and hybrid or semi-supervised approaches where both types are combined prevail. Most of machine learning approaches (except neural networks) rely on so called feature engineering. ...

Hidden Markov Models (HMM) are finite-state automatons designed for modeling hidden sequences based on some observations. Basic assumption of HMM is that following state depends only on the current state. For each state there are defined probabilities of transition to another state and each state emits an observation with certain probability.

We can formally define HMM as following tuple (patica ??):

$$M = (Q, A, O, B, \pi) \tag{1.1}$$

where:

- Q is the set of all possible states, $Q = q_1, q_2...q_n$
- A is the matrix of transition probabilities $A = a_{11}...a_{ij}...a_{nn}$, a_{ij} is the probability of transition from state i to state j, therefore $\sum_{j=1}^{n} a_{ij} = 1, \forall i$
- O is the sequence t of observations from vocabulary $V = v_1, v_2...v_v$, $O = o_1, o_2...o_t$
- B is the probability of emitting observation $B = b_i(o_j)$ is probability

of emitting observation o_j in state i

- π is the initial distribution of probabilities, $\pi = \pi_1, \pi_2 ... \pi_n$, π_i is the probability that initial state will be i, the following equations has to hold: $\sum_{i=1}^{n} \pi_i = 1$

In case of NER states are named entities and emitted symbols are words (more precisely: features of words). Baum Welch algorithm is used for training probabilities and maximize likelihood of emitting observations from train set by model M ($P(O|M)$). In inference we have observations - words - and we are looking for most probable sequence of states in automaton that generates given observations. This can be done by dynamic programming - Viterbi alghoritm. Found sequence are NER tags.

Another frequently used method for labeling sequences which is similar to HMM are **Conditional Random Fields (CRF)**. One of the most important difference between them is that CRF is a discriminative model opposed to generative HMM. Joint distribution $P(Y|X)$ is used and thus it should work better with complex and dependant features. Despite the fact that this approach was used in early years of NER it is still used nowadays in NER pipelines in combination with BiLSTM neural networks

Maximum Entropy Models (MEM) are also discriminative, but opposite to CRF and HMM they don't take in account order of sequence. Their corner-stone are feature functions, which can look like this:

$$f_1(x, y) \begin{cases} 1 \; if \; x \; is \; verb \; and \; y \; is \; PERSON \\ 0 \text{ otherwise} \end{cases} \tag{1.2}$$

where x is a words and y is it's ner tag. This functions models given dataset by learning statistical information about it and follows Maximum Entropy principle which means that from all models that satisfy learned information about dataset it will choose one with maximum entrypy. MEM models are defined as follows:

$$P(y|x) = \frac{1}{Z(x)} exp(\sum_{i=1}^{n} w_i f_i(x, y)) \tag{1.3}$$

$$Z(x) = \sum_{y'} exp(\sum_{i=1}^{n} w_i f_i(x, y)) \tag{1.4}$$

where w_i is weight of feature function and $Z(x)$ just helps to normalize probabilities. For inference MEM computes probability for each tag and takes one with highest.

While all machine learning approaches we've already mentioned are relatively similar, the next one **Support Vector Machines (SVM)** differs. SVMs are binary classifiers that try to learn optimal hyperplane that separates positive and negative samples from input data and thus classify them. It maximizes margin between hyperplane and so called *support vectors* - points from input dataset that are closest to the plane. Then a prediction for point x is done as follows:

$$class(x) = \begin{cases} positive\ if\ wx + b > 0 \\ negative\ if\ wx + b < 0 \end{cases} \tag{1.5}$$

where w and b are learned parameters of separating hyperplane. From this explanation it's clear that SVMs would be able to separate classes only if input data were linearly separable. When this is not true SVMs uses *kernel trick* which allows easy transformation of data into higher-dimensional space

where they might be separable.

Although SVMs are binary classifiers there are methods how to use them for multiple classes (e.g. train binary classifier for each pair of classes and then combine them with voting). This allows to use them successfully also for NER.

Neural Networks (NN) After 2010 first NN models were introduced for Named Entity Recognition. They were mostly based on Recurrent Neural Networks (RNN) which are designed for sequential data or in minority on Convolution Neural Network (CNN). The big advantage of NN over previous approaches is that they require minimal feature engineering or some hand made data like dictionaries. Therefore it is easier to develop them and they are less domain or language dependant.

1.2.3 Recurrent Neural Networks

As it was previously written purpose of RNN is to deal with sequential data. It processes input sequence x $(x(1), ...x(T))$ in order and keeps hidden state which aggregates information from previous elements. The most simple RNN architecture consist of 3 layers: input layer x, hidden layer h and output layer y. They are computed as follows:

$$\begin{aligned} h_t &= f_h(W_x x(t) + W_h h_{t-1}) \\ y_t &= f_y(W_y h_t) \end{aligned} \tag{1.6}$$

W_x, W_h and W_y are weight matrices and f_h, f_y are activation functions. Most common activation functions are $sigmoid$, $tanh$, $ReLU$ or $softmax$ for output layer. For training are usually used extensions of backpropagation algorithm called Backpropagation through time or Real time recurrent

learning. Simple illustration of how RNN process sequential data is shown in 1.3.

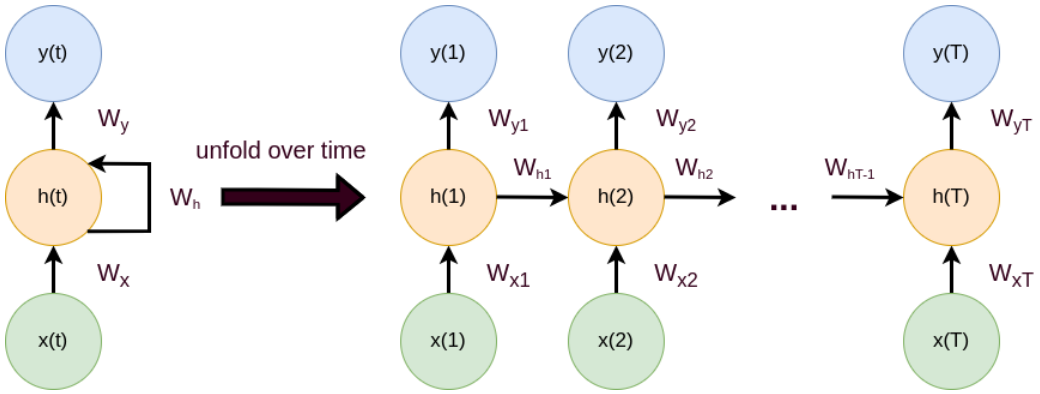


Figure 1.3: Illustration of RNN architecture

However RNNs suffer of problem with vanishing or exploding gradient. It occurs during training as we multiply matrices over and over. The gradient values start to decrease until disappearing completely and thus previous states stop to contribute to current step computation. Opposite problem is exploding gradient when values increase too much and influence current step updates too heavily.

As a solution to this problem were introduced 'gated neurons' Long Short-Term Memory Units (LSTM) and Gated Recurrent Units (GRU). They control flow of data from input and hidden state allowing network to propagate important data further and forget unimportant. So called 'LSTM cell' can be seen in 1.4. Input c_t that we haven't seen before in RNN architecture is cell state and it's another state that cell keeps and updates thought time.

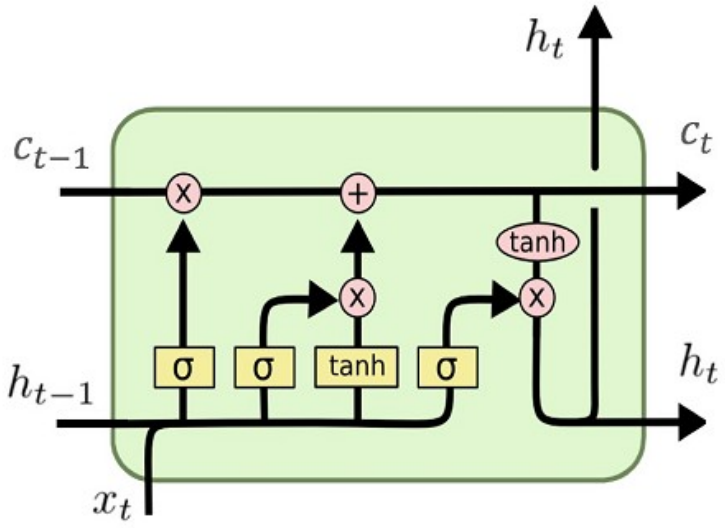


Figure 1.4: LSTM cell [8]

LSTM RNN we have seen so far can be successfully used for sequence labeling task as is NER. As it goes through sentence $S = s(1)...s(T)$ on output it produces labels for each word while having information about all words that precedes current word. However sometimes it is useful to look in 'the future' at following words since they can also contain information about previous words. For example in sentence *Today i saw Buckingham palace.* we don't know if label of *Buckingham* should be Person or Location till we see the word *palace*. To resolve this issue **Bidirectional RNNs** were introduced. The main idea behind them is that we use 2 LSTM layers, one processing sequence from the beginning and second one from the end and conjugate their outputs. An illustration of such network can be seen in 1.5 (todo: word level vs character level modeling, word embeddings)

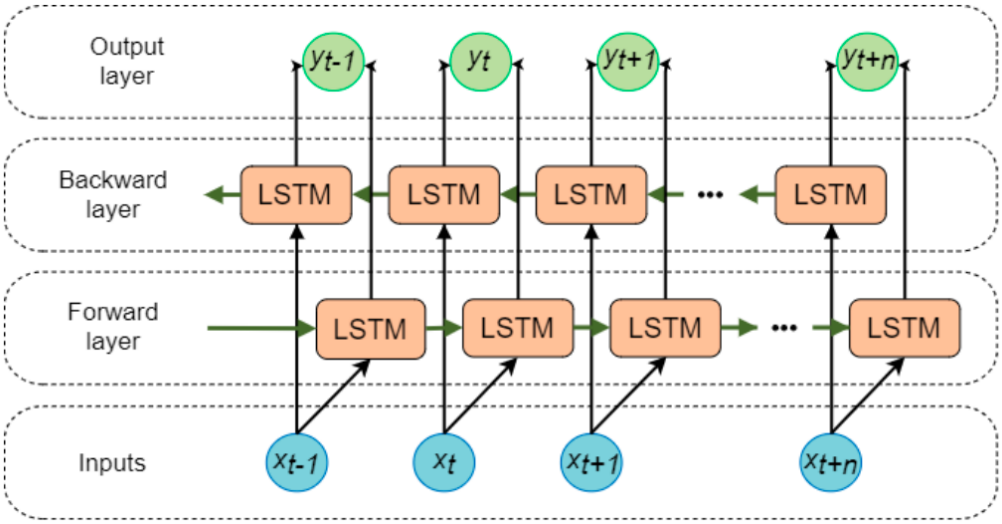


Figure 1.5: Bi-LSTM RNN [9]

BiLSTM architectures were successfully used for Named Entity Recognition. The best results were achieved (todo: source) when there was CRF model trained on the top of the BiLSTM neural network.

Another important step in RNN design are **Encoder - Decoder** architectures. An illustration of such network using LSTMs is shown in 1.6. This architecture consists of two separate networks. First one - Encoder is responsible for encoding input sequence into some state (last hidden state) which is then passed to second part - Decoder, which generates from it output sequence. The output sequence can be fixed length (doesn't have to be the same length as input one) or until special token that signalizes end of sequence is generated. Since this model takes as an input one sequence and produces another it belongs to a class of models called **Sequence to Sequence (Seq2Seq)**. It dominates in several NLP tasks such as machine translation, speech recognition or question answering. Although this architecture is also used for NER it isn't the most popular one. However it is important part of more complex model which made significant breakthrough

in many NLP tasks in recent years - **Transformers**.

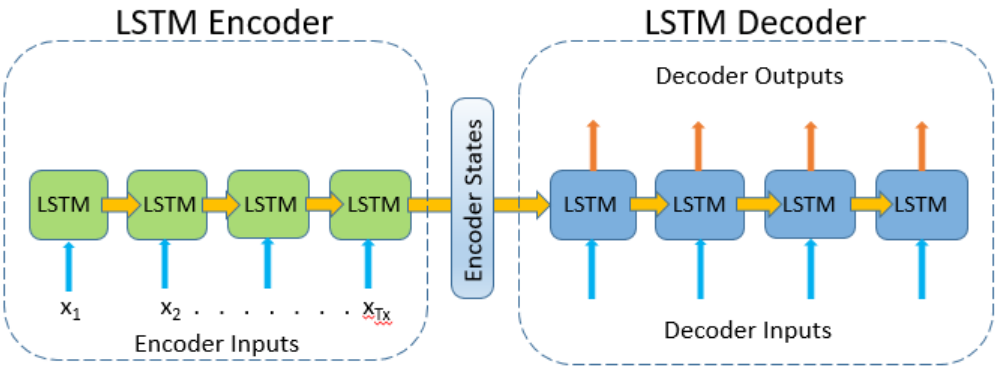


Figure 1.6: Encoder Decoder architecture [10]

1.2.4 Transformers

Although objective of transformer model is to process sequences and it's Encoder-Decoder it completely replaces RNN design of sequential computation of hidden state with so called Attention layer. It consists of three vectors - Query (Q), Key (K) and Value (V) and it is computed as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1.7}$$

d_k is length of K vector and it's only for regularization. An attention is computed for each word in a sequence and abstraction over it is that it allows model to focus (pay attention) to all other words in a sequence while processing specific word and evaluate their importance for current word. Q, K, V are computed by multiplying input vectors with matrices W_Q, W_K, W_V which are trained during training. Query vector can be seen as representation of current word, Key vector as an contribution of other words and Value as how they contribute. Since there is no sequential information like in RNNs,

positional encoding is used for encoding words into vectors. An example of such a function is shown below.

$$\begin{aligned} PE(pos, 2i) &= sin(pos/10000^{2i/d_{model}}) \\ PE(pos, 2i+1) &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{1.8}$$

An overview over the whole transformer architecture is at 1.7. Components of transformer Encoder:

- **Multi-Head Attention** - performs attention computation multiple times in parallel with different weights and concatenate them afterwards. It should allow model to gather information from different positions and subspaces.
- **Add & Norm** - It adds residual connection with output of previous layer and normalize it.
- **Feed Forward NN** - transforms data for the next layer.

Components of transformer Decoder (that are not in Encoder):

- **Masked Multi-Head Attention** - The only difference is that only words that are already generated are accessible - attention cannot look in to the 'future'.
- **Linear** - Also feed-forward network, which maps output of previous layers to dimension of vocabulary.
- **Softmax** - Transforms outputs to probabilistic interpretation.

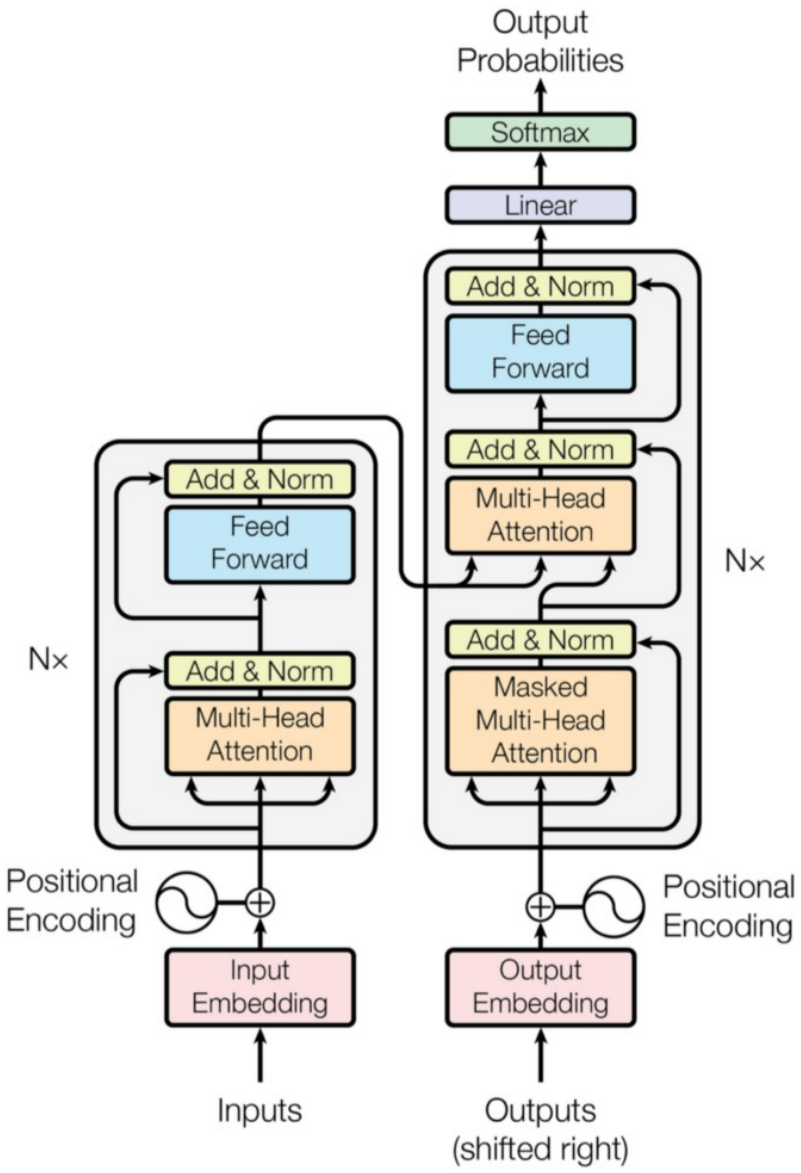


Figure 1.7: Transformer architecture [11]

The transformer architecture is a base of new class of language models. They are called pre-trained deep contextualized language models and they took over most of NLP tasks in the last years.

The idea behind them is to use unsupervised training on large amount of data

in order to learn underlying structures in natural language. Afterwards only not as resource demanding finetuning is needed to adjust model for specific task.

1.3 Related work

In this section we will first discuss the state of a NER task in slovak texts, used methods and achieved results. Secondly we will discuss the state-of-the-art approaches for low resource languages such as Slovak language.

1.3.1 State of the NER for Slovak texts

There are not many works that addresses slovak NER. It is a task that needs extensive datasets - which were and still are - missing. Because of the lack of data different approaches based on rules and vocabularies were used. Authors of [12] created system using multiple dictionaries, regular expressions and multiword expressions. Since there was no test dataset available at the time, it wasn't possible to evaluate the system. In [13] two step system was developed. In the first step text was preprocessed - stemmed. In the next step potential entities - words with capital letters - were compared with database of already recognized entities. New entities were recognized by finding entity scope through Wikipedia parsing. For purpose of this work dataset of 60 news article was annotated by human experts with total of 1620 entities. Result of 79% F-score was obtained. Unfortunately neither the dataset nor the exact method was published and therefore new methods cannot be compared with this one.

However in the last years several master thesis on multiple slovak universities focused on a NER with machine learning and deep learning approaches.

In [14] authors created dataset from more than 5000 articles downloaded from slovak Wikipedia containing more than 15 000 entities. For training the model they used popular python NLP library Spacy and achieved F-score a little bit over 72%. Authors of [15] worked with similar sized dataset and have tried NER pipelines from multiple production ready NLP libraries and achieved 69% accuracy of detecting named entities and 78% accuracy of categorizing entities. Their overall F-score was 73%.

Also in following master thesis [7] authors have decided to create their own dataset. As a source of the texts they have chosen a twitter account of one of the biggest slovak journal. Through crawling they gathered 22 thousand tweets but annotated only 10 thousand with almost 16 thousand entities. Datset was divided to train and two test sets in ration 8:1:1. Firstly words were encoded to vectors with FastText library and subsequently recognized and categorized by biLSTM recurrent neural network and achieved macro F1-score of 91.36%. Authors also trained their model on standard english NER dataset CoNLL2003 and thus compared their model architecture with cutting edge solutions. In these experiments competitive results were obtained that are showing that proposed model is state-of-the-art. Similar to previous works neither the dataset nor the trained model were published.

Chapter 2

Research

2.1 WikiANN

Since WikiANN is the only dataset publicly available for NER in Slovak language, we've started experimets wit it. In table 2.1 we can see results of 4 models trained on this dataset. We used most common python NLP libraries that support Named Entity Recognition task.

- **Trankit** - It's lightweight transformer based NLP library that has pipeline 'custom-ner' which allowed us to train model for Slovak NER.
- **Spacy** - Spacy is complex and effective production ready library focusing solely on NLP. We have trained 2 models with different word embeddings.
- **HuggingFace - Transformers** - Another library targeting simple work with transformer models. It provides many pre-trained and easy to use models.

	Precision	Recall	F1
Trankit	88.9	88.51	88.7
Spacy: tok2vec + Transition-based parsing	84.49	82.96	83.72
Spacy: Multilingual Bert + Transition based parsing	92.64	92.33	92.49
Transformers: SlovakBert	91.4	92.8	92.1

Table 2.1: Results of several NER models trained on WikiANN dataset

Chapter 3

Results

Bibliography

[1] Elizabeth D. Liddy. Natural language processing. 2001.

[2] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 08 2007.

[3] Archana Goyal, Vishal Gupta, and Manish Kumar. Recent named entity recognition and classification techniques: A systematic review. *Computer Science Review*, 29:21–43, 2018.

[4] Machine Learning. An introduction to conditional random fields. *Mach. Learn*, 4(4):267–373, 2011.

[5] Girish Palshikar. *Techniques for Named Entity Recognition: A Survey*, volume 1, pages 191–. 01 2012.

[6] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[7] Jakub Mičo. Rozpoznávanie pomenovaných entít metódami strojového učenia, 2019. Slovenská technická univ. v Bratislave FIIT.

[8] https://colah.github.io/posts/2015-08-understanding-lstms.

[9] Delanyo KB Kulevome, Hong Wang, and Xuegang Wang. A bidirectional lstm-based prognostication of electrolytic capacitor. *Progress In Electromagnetics Research C*, 109:139–152, 2021.

[10] https://towardsdatascience.com/time-series-forecasting-with-deep-stacked-unidirectional-and-bidirectional-lstms-de7c099bd918.

[11] http://jalammar.github.io/illustrated-transformer.

[12] Marek Suppa and Ondrej Jariabka. Benchmarking pre-trained language models for multilingual NER: TraSpaS at the BSNLP2021 shared task. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 105–114, Kiyv, Ukraine, April 2021. Association for Computational Linguistics.

[13] Kaššák Ondre, Kompan Michal, and Mária Bieliková. Sextrakcia pomenovaných entít pre slovenský jazyk. In *ZNALOSTI 2012 : Sborník příspěvků 11 ročníku konference, Mikulov, hotel Eliška*, pages 52–61. Praha: Matfyzpres, 2012.

[14] Jakub Maruniak. Anotácia a rozpoznávanie pomenovaných entít v slovenskom jazyku, 2021. Technická univerzita v Košiciach.

[15] Dávid Lupták. Rozpoznávanie pomenovaných entít metódami strojového učenia, 2021. Technická univerzita v Košiciach.

[16] Afshin Rahimi, Yuan Li, and Trevor Cohn. Massively multilingual transfer for ner, 2019.

[17] Ján Staš, Daniel Hládek, Stanislav Ondáš, Daniel Zlacký, and Jozef Juhár. Spracovanie prirodzeného jazyka pre interaktívne rečové rozhrania v slovenčine. In *ITAT*, pages 81–87, 2015.

[18] Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. A survey on recent approaches for natural language processing in low-resource scenarios, 2021.

List of Figures